



**UNIVERSIDAD TECNOLÓGICA DE SANTIAGO, UTESA**

**SISTEMA CORPORATIVO**

**Facultad de Ingeniería y Arquitectura**

**Carrera de Ingeniería en Sistemas Computacionales**

**Programación de Videojuegos**

**Presentado por:**

Jheyson Toribio, 1-16-0119

Diógenes Fernández,

**Presentado a:**

Ing. Iván Mendoza

**Santiago de los caballeros**

**República Dominicana**

**Diciembre, 2021**

## **INTRODUCCIÓN**

### **CAPÍTULO I: VIDEOJUEGO Y HERRAMIENTAS DE DESARROLLO**

#### **1.1 Descripción**

KeyHero es un juego de acción con música, donde se desafían los sentidos auditivos y rítmicos de los usuarios, requiriendo que los mismos toquen botones de una secuencia en pantalla para crear los ritmos esperados.

#### **1.2 Motivación**

La principal motivación para hacer este juego fue la curiosidad de probar que tan rápido pueden reaccionar los sentidos del humano ante ciertos dictados orientados a un resultado específico. Juegos como Magic Tiles y BeatStar fueron inspiración.

##### **1.2.1 Originalidad de la idea**

Un dato característico del juego que lo hace diferente al resto es la adaptación a un estilo propio, ideado por Diógenes, quien aparte de recibir influencias de los videojuegos jugados, también tiene de su pasión por la música y algunos instrumentos.

##### **1.2.2 Estado del Arte**

Es un juego desarrollado en 2D, un tipo de juego desarrollado por una gran cantidad de empresas.

#### **1.3 Objetivo general**

Crear un ritmo completo con melodías dictadas.

#### **1.4 Objetivos específicos**

- Tocar todas las melodías sin dejar pasar alguna
- Tocar las melodías en el orden indicado

#### **1.5 Escenario**

El juego consta de un escenario de fondo negro, con 4 botones de colores que y flechas que irán bajando para indicar a que botón se le debe tocar a su llegada.

#### **1.6 Contenidos**

Se presenta un videojuego desarrollado en 2D, donde el usuario toca botones para crear un ritmo, de acuerdo dictados en pantalla.

#### **1.7 Metodología**

##### **1.7.1. Selección de herramientas**

Se escogió la herramienta de Unity para desarrollar el videojuego ya que es práctica e interactiva además proporciona una facilidad de uso.

### 1.7.2. Documentación

Hubo asistencia de diferentes escritos y videotutoriales para la construcción del videojuego, centrándonos en las funciones específicas que el juego tendrá.

### 1.7.3. Definición del juego

Se decidió hacer un juego en el que el usuario toque un botón para crear una melodía, la cual en conjunto con el resto crearía un ritmo final, en caso de que el usuario logre completarlo.

## 1.8 Arquitectura de la aplicación

La aplicación está diseñada para ejecutarse de manera web, y también para desktop, con planes de en un futuro ser llevada a la versión móvil

## 1.9 Herramientas de desarrollo

- Unity
- Adobe Suite
- Visual Studio

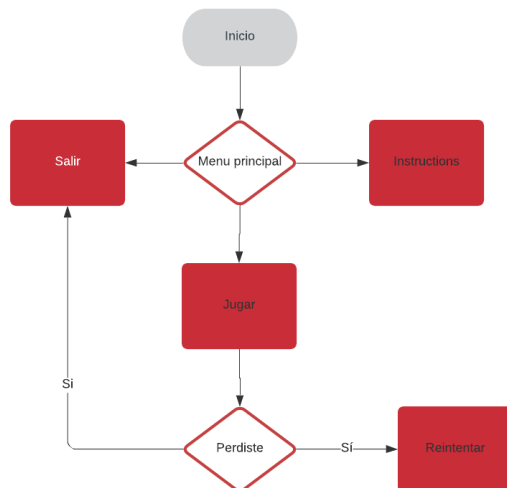
## CAPÍTULO II: DISEÑO E IMPLEMENTACIÓN

### 2.1 Planificación (Diagrama de Gantt)

	Semana 1	Semana 2	Semana 3	Semana 4
Capítulo 1	Jheyson y Diógenes			
Capítulo 2	Diógenes y Jheyson			
Capítulo 3			Diógenes y Jheyson	
Capítulo 4			Jheyson y Diógenes	

Leyenda:  Completado  Pendiente

### 2.2 Diagramas y Casos de Uso



## 2.3 Plataforma

Desktop y web

## 2.4 Género

Acción

## 2.5 Clasificación

Everyone (E)

## 2.6 Tipo de Animación

2D

## 2.7 Equipo de Trabajo

- Productor: Diógenes Fernández y Jheyson Toribio
- Diseñador: Diógenes Fernández y Jheyson Toribio
- Artista: Diógenes Fernández
- Programador: Diógenes Fernández
- Sonido: Diógenes Fernández y Jheyson Toribio
- Aseguramiento de calidad: Diógenes Fernández y Jheyson Toribio
- Ventas y mercadotecnia: Jheyson Toribio
- Administración de la comunidad: Jheyson Toribio

## 2.13 Mecánica del Juego

KeyHeroes funciona presionando un botón al momento de que una melodía venga hacia él. al presionarlo se estaría creando una melodía, que en un conjunto con el resto crearían un ritmo, que es el objetivo del juego.

## CAPÍTULO III: DESARROLLO

### 3.1 Capturas de la Aplicación (Documentación completa del desarrollo, Scripts, Sprites, Prefabs e imágenes)



**Usa Las Teclas Del  
Teclado Para Jugar**



F



G



H



J



## **LISTA DE MUSICA**

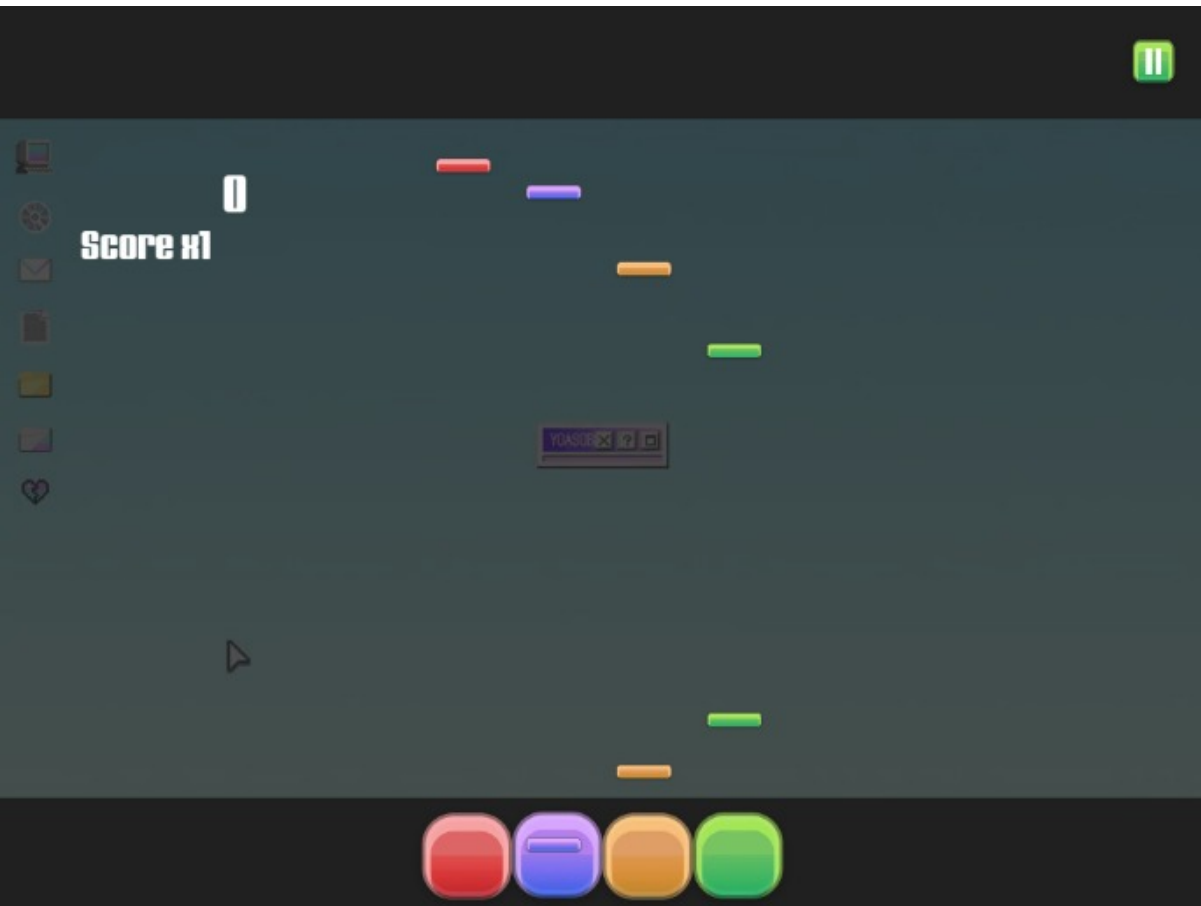
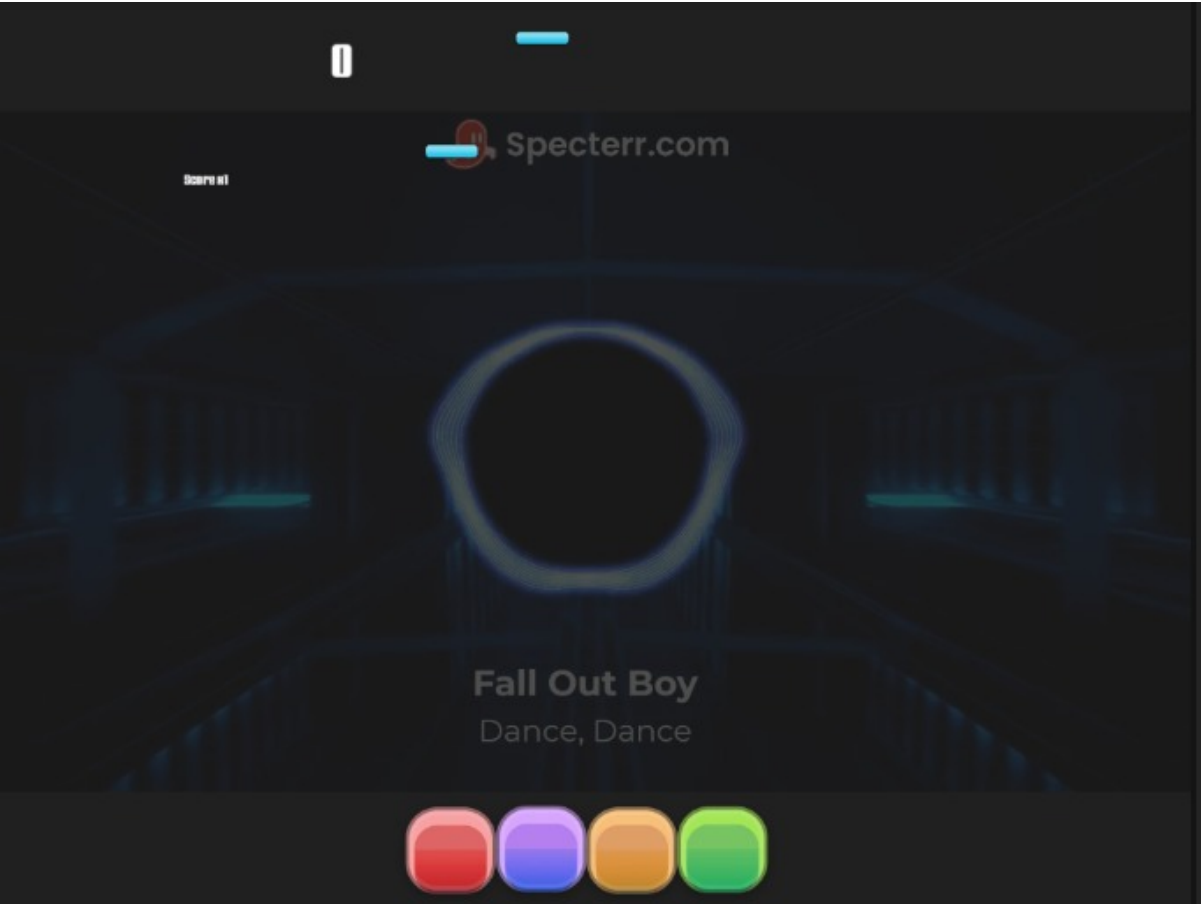
**YOASOBI - Halcyon**  
**Fall Out Boy - Dance**



## **LISTA DE MUSICA**

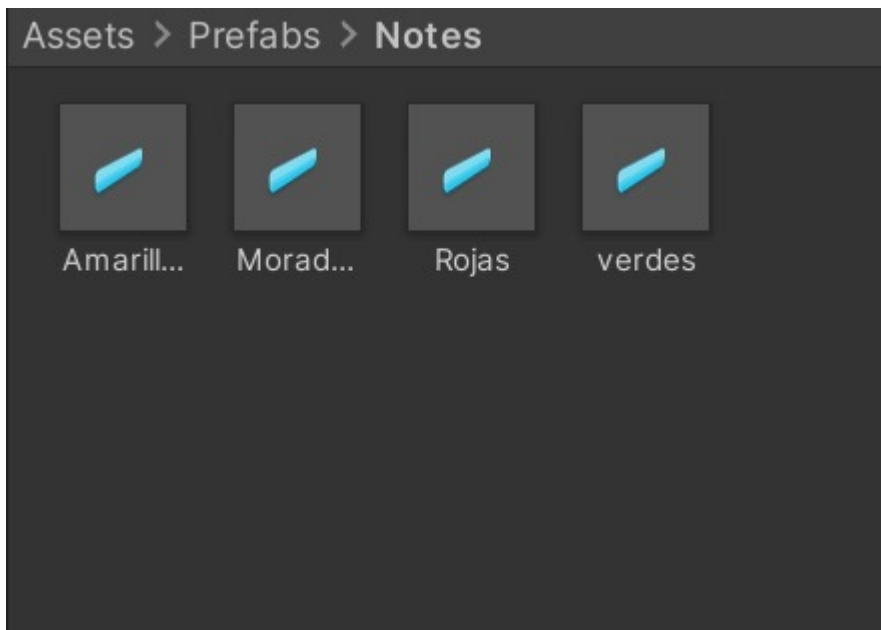
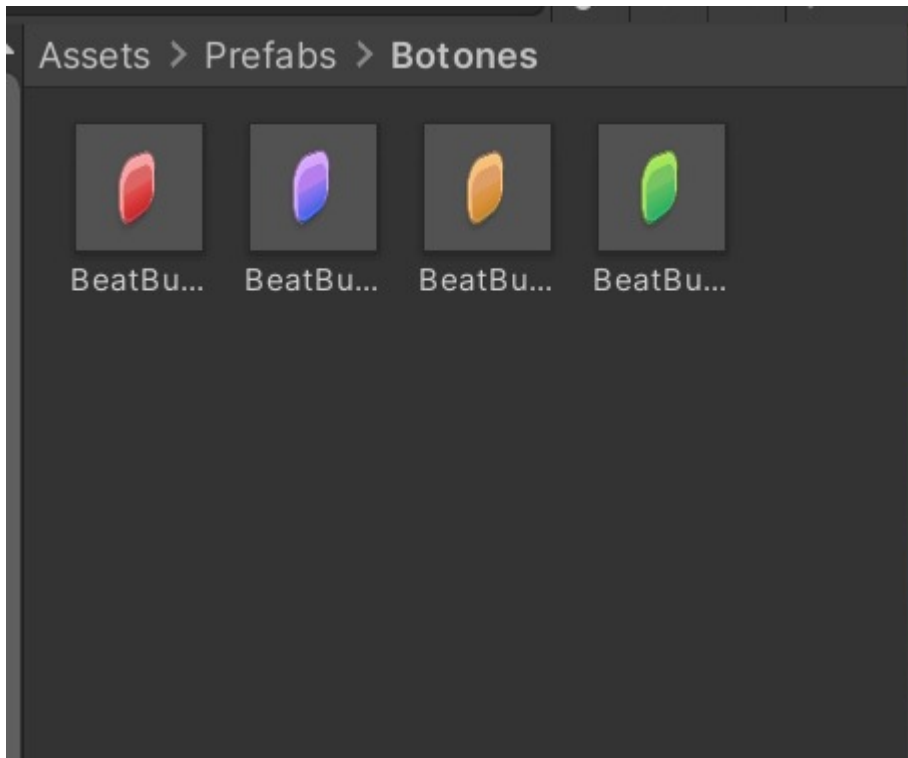
**YOASOBI - Halcyon**  
**Fall Out Boy - Dance**











```
BeatScroller.cs* X
C# Archivos varios BeatScroller

1  using System.Collections;
2  using System.Collections.Generic;
3  using System.Collections.Specialized;
4  using System.Security.Cryptography;
5  using System.Threading;
6  using UnityEngine;
7
8  public class BeatScroller : MonoBehaviour
9  {
10
11     public bool createMode;
12     public float beatTempo;
13     public bool hasStarted;
14     public GameObject n;
15     public KeyCode keyToPress;
16
17
18     // Start is called before the first frame update
19     void Start()
20     {
21
22
23
24         //Para el modo creador
25         //Esto pone en reversa el movimiento de la lista
26         if(createMode)
27         {
28             beatTempo = beatTempo / -60f;
29         }
30         else{
31             beatTempo = beatTempo / 60f;
32         }
33
34     }
35
```

```
BeatScroller.cs* - X
Archivos varios BeatScroller

34     }
35
36     // Update is called once per frame
37     void Update()
38     {
39
40         if (!hasStarted)
41         {
42             if (Input.anyKeyDown)
43             {
44                 hasStarted = true;
45             }
46         } else
47         {
48
49             //Para el modo creador
50             //Esto pone una nota cada vez que se presiona un boton
51             if(createMode)
52             {
53
54                 if(Input.GetKeyDown(keyToPress))
55                 {
56
57                     Instantiate(n,transform.position,Quaternion.identity);
58                 }
59                 transform.position -= new Vector3(0f, beatTempo * Time.deltaTime, 0f);
60             }
61             //Si el modo creador esta activado
62             //Esto es para el movimiento del mapa de notas
63             else
64             {
65                 transform.position -= new Vector3(0f, beatTempo * Time.deltaTime, 0f);
66             }
67         }
68     }
69 }
```

```
ButtonController.cs  BeatScroller.cs*
Archivos varios  ButtonController

1  using System.Collections;
2  using System.Collections.Generic;
3  using System.Collections.Specialized;
4  using System.Security.Cryptography;
5  using System.Threading;
6  using UnityEngine;
7
8  public class ButtonController : MonoBehaviour
9  {
10
11     private SpriteRenderer theSR;
12     public Sprite defaultImage;
13     public Sprite pressedImage;
14     public KeyCode keyToPress;
15
16
17     void Start()
18     {
19         //inicia el renderizador de sprites
20         theSR = GetComponent<SpriteRenderer>();
21     }
22
23
24     // Update is called once per frame
25     void Update()
26     {
27
28         //si se preciona el boton pone la imagen de precionado
29         if (Input.GetKeyDown(keyToPress))
30         {
31             theSR.sprite = pressedImage;
32         }
33         //Obtiene la tecla del teclado a precionar
34         if (Input.GetKeyUp(keyToPress))
35         {
```

```
GameManager.cs ButtonController.cs BeatScroller.cs*
Archivos varios GameManager
1 using System.Collections;
2 using System.Collections.Generic;
3 using System.Diagnostics;
4 using System.Runtime.InteropServices;
5 using UnityEngine;
6 using UnityEngine.UI;
7
8 public class GameManager : MonoBehaviour
9 {
10
11     public AudioSource theMusic;
12     public AudioSource beatSfx;
13
14     public bool startPlaying;
15
16     public BeatScroller theBS;
17
18     public static GameManager instance;
19
20     public int currentScore;
21     public int scorePerNote = 2000;
22     public int scorePerGoodNote = 2500;
23     public int scorePerPerfectNote = 3000;
24
25     public int currentMultiplier;
26     public int multiplierTracker;
27     public int[] multiplierThresholds;
28
29     public Text scoreText;
30     public Text multiText;
31
32     public float totalNotes;
33     public float NormalHits;
34     public float GoodHits;
35     public float PerfectHits;
```

```
GameManager.cs x ButtonController.cs BeatScroller.cs*
Archivos varios GameManager
31
32 public float totalNotes;
33 public float NormalHits;
34 public float GoodHits;
35 public float PerfectHits;
36 public float MissedHits;
37
38 public GameObject resultsScreen;
39 public Text percentHitText, normalsText, goodsText, perfectsText, missesText, rankText, finalScoreText;
40
41
42
43 // Start is called before the first frame update
44 // Para iniciar la partida
45 void Start()
46 {
47     instance = this;
48     scoreText.text = "Preciona Una Tecla Para Empezar";
49
50     currentMultiplier = 1;
51     totalNotes = ((FindObjectsOfType<NoteObject>()).Length) - 168;
52 }
53
54 // Update is called once per frame
55 void Update()
56 {
57     if (!startPlaying)
58     {
59         //Detecta cuando se preciona una tecla para iniciar
60         if (Input.anyKeyDown)
61         {
62             scoreText.text = "0";
63             startPlaying = true;
64             theBS.hasStarted = true;
65             theMusic.Play();
66         }
67     }
68 }
```

```
GameManager.cs x ButtonController.cs BeatScroller.cs*
Archivos varios GameManager
67 }
68 else
69 {
70     //muestra el resultado al finalizar la musica
71     // Y la calificacion Desde A hasta F
72     if ((!theMusic.isPlaying /*|| !(theMusic.time != 0)*/) && (!resultsScreen.activeInHierarchy))
73     {
74         resultsScreen.SetActive(true);
75         normalsText.text = "" + NormalHits;
76         goodsText.text = "" + GoodHits;
77         perfectsText.text = "" + PerfectHits;
78         missesText.text = "" + MissedHits;
79
80         float totalHits = NormalHits + GoodHits + PerfectHits;
81         float percentHit = (totalHits / totalNotes) * 100f;
82
83         percentHitText.text = percentHit.ToString("F1") + "%";
84
85         string rankVal = "F";
86
87         if (percentHit >= 30)
88         {
89             rankVal = "D";
90             if (currentScore >= 550000)
91             {
92                 rankVal = "C";
93                 if (currentScore >= 750000)
94                 {
95                     rankVal = "B";
96                     if (currentScore >= 900000)
97                     {
98                         rankVal = "A";
99                         if (currentScore >= 950000)
100                         {
101                             rankVal = "S";
102                         }
103                     }
104                 }
105             }
106         }
107     }
108 }
```



```
GameManager.cs ButtonController.cs BeatScroller.cs*
Archivos varios GameManager
100 {
101     rankVal = "S";
102 }
103 }
104 }
105 }
106 }
107 rankText.text = rankVal;
108 finalScoreText.text = currentScore.ToString();
109 }
110 }
111 }
112 //Esto calcula los puntos de al precionar las notas
113 //Desde aqui hasta el final segun la precision del jugador
114 public void NoteHit()
115 {
116     beatSfx.Play();
117     multiplierTracker++;
118     if ((currentMultiplier - 1) < multiplierThresholds.Length)
119     {
120         multiplierTracker++;
121         if (multiplierThresholds[currentMultiplier - 1] <= multiplierTracker)
122         {
123             multiplierTracker = 0;
124             currentMultiplier++;
125         }
126     }
127     scoreText.text = "" + currentScore;
128     multiText.text = "Score x" + currentMultiplier;
129 }
130 }
131 }
132 }
133 }
134 public void NormalHit()
135 {
```

```
GameManager.cs ButtonController.cs BeatScroller.cs*
Archivos varios GameManager
133
134 public void NormalHit()
135 {
136     UnityEngine.Debug.Log("Normal Hit");
137     currentScore += scorePerNote * currentMultiplier;
138     NormalHits++;
139     NoteHit();
140 }
141
142 public void GoodHit()
143 {
144     UnityEngine.Debug.Log("Good Hit");
145     currentScore += scorePerGoodNote * currentMultiplier;
146     GoodHits++;
147     NoteHit();
148 }
149
150 public void PerfectHit()
151 {
152     UnityEngine.Debug.Log("Perfect Hit");
153     currentScore += scorePerPerfectNote * currentMultiplier;
154     PerfectHits++;
155     NoteHit();
156 }
157
158 public void NoteMissed()
159 {
160     UnityEngine.Debug.Log("missed");
161     MissedHits++;
162     currentMultiplier = 1;
163     multiplierTracker = 0;
164     multiText.text = "Score x" + currentMultiplier;
165 }
166
167
```

# KEYHEROES

Play

Instructions

Options

Quit

```
SongManager.cs  X  GameManager.cs  ButtonController.cs  BeatScroller.cs*
Archivos varios  SongManager

7  using System;
8
9  public class SongManager : MonoBehaviour
10 {
11     public static SongManager Instance;
12     public AudioSource audioSource;
13     public float songDelayInSeconds;
14     public double marginOfError; // in seconds
15     public int inputDelayInMilliseconds;
16     public string fileLocation;
17     public float noteTime;
18     public float noteSpawnY;
19     public float noteTapY;
20
21     public float noteDespawnY{
22         get
23         {
24             return noteTapY - (noteSpawnY - noteTapY);
25         }
26     }
27
28     //Inicia la cancion del juego
29     public void StartSong()
30     {
31         audioSource.Play();
32     }
33     //Busca el tiempo que dura el audio de la musica para detener el juego al final
34     public static double GetAudioSourceTime()
35     {
36         return (double)Instance.audioSource.timeSamples / Instance.audioSource.clip.frequency;
37     }
38
39     void Update()
40     {
41
```

### **3.2 Prototipos**

- Primer prototipo tocando cualquier botón para crear el mapa de piezas en el nivel
- Segundo prototipo con el primer nivel creado
- Tercer prototipo con Menues e instrucciones
- Cuarto prototipo Con el segundo nivel creado
- Quinto prototipo Prueba de Build para PC (Aprobado)
- Sexto prototipo Prueba para Build en Web (Aprobado con errores)
  - No carga los videos a menos que esté en pantalla completa

### **3.3 Perfiles de Usuarios**

- Personas mayores de 10 años de edad
- Apasionados por la música
- Personas que le guste poner a prueba sus sentidos

### **3.4 Usabilidad**

- Utilizar las teclas del teclado
- Tocar los botones de acuerdo a la melodía que viene

### **3.5 Test**

El público elegido para hacer la prueba del juego comprende entre 10 y 25 años y les gustó, pero recomendaron adaptarlo a música tradicional dominicana, como el merengue y la bachata.

### **3.6 Versiones de la Aplicación**

- Versión 0.6 : Música Electrónica y Rock , ideal para el juego, con 4 botones.

## **CAPÍTULO IV: PUBLICACIÓN**

### **4.1 Requisitos de la instalación**

No requiere instalación, ya que será un archivo ejecutable, o lo que llamamos una versión portable del sistema. Sin embargo, para un funcionamiento adecuado se necesita:

- En escritorio
  - 100 MB disponible de espacio en disco
  - 1 GB de RAM o más
- En la web
  - Flash Player

### **4.2 Instrucciones de Uso**

Para seleccionar un botón en este videojuego, deberás presionar una de las siguientes teclas:

F: Izquierdo

G: Izquierdo interno

H: Derecho interno

J: Derecho

### 4.3 Bugs

El proyecto final no ha presentado bugs hasta el momento. Habiendo pasado todas las pruebas realizadas, tenemos certeza al decir que tenemos un sistema libre de bugs.

### 4.4 Proyección a Futuro

Se está trabajando para posteriormente llevar el juego a los móviles con mejoras aplicadas y una mejor experiencia para el usuario. La idea es que pueda usarse tanto en Android como en iOS.

### 4.5 Presupuesto

DC: Diseñador Creativo - RD \$25,000

P: Programador - RD \$25,000

UT: Usuario Tester - RD \$12,000

TS: Técnico de Sonido - RD \$20,000

	DC	P	UT	TS
Tiempo	1 mes	1 mes	1 semana	25 días
Costo (RD \$)	25,000	25,000	12,000	20,000

### 4.6 Análisis de Mercado

En los últimos 3 años, la industria de los videojuegos ha sumado 500 millones de jugadores, alcanzando así unos 2,700 millones en todo el mundo, y se esperan unos 400 millones más para finales del 2023.

Según informes, el 60% de nuevos jugadores son mujeres, y un 30% son menores de 25 años. Sin embargo, de los llamados expertos jugando, el 61% está compuesto por varones.

### 4.7 Viabilidad

El videojuego se ejecutará en ordenador y en la web, y no necesita pago.

## **CONCLUSIONES**

Al concluir este videojuego se crea una barrera de tiempo en la República Dominicana, ya que hasta donde se investigó, no se ha desarrollado ninguno de ese tipo anteriormente en dicho territorio. Al pasar el tiempo será adaptado a música tradicional del país, por lo que será mucho más icónico.

Con dicho proyecto se busca poner a prueba los sentidos humanos, y más que buscarlo, se logra y se logró en cada prueba. Queda al disfrute del usuario!



## REFERENCIAS BIBLIOGRÁFICAS

- Unity: <https://unity.com/>
- Manual de Unity: <https://docs.unity3d.com/es/530/Manual/UnityManual.html>
- Game: <https://drive.google.com/drive/folders/11nF5k-DFhsV8YHqOp0VfBsdMdHE-j1QX>
- Itch: <https://dingoarts.itch.io/keyheroes>